

Assignment 3

目录

1. 质量属性效用树.....	2
2. 构架方法分析.....	2
3. 敏感点与权衡点列表.....	7
4. 有风险决策与无风险决策列表.....	8
5. 个人总结.....	9

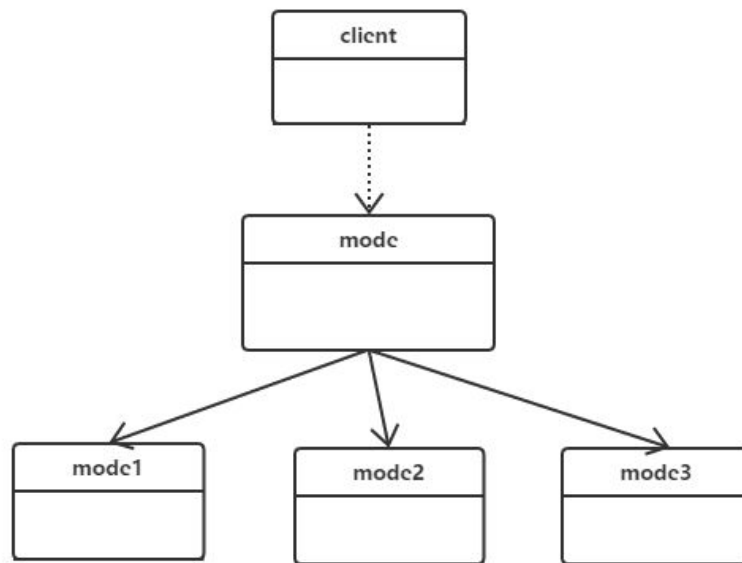
1.质量属性效用树

质量属性	属性求精	场景
可扩展性		A1-添加自定义获取模式(H,M)
		A2-添加外部数据格式(H,L)
		A3-添加外部设备支持(H,M)
可用性		A4-检测远程图像处理服务器故障并从中恢复过来(H,M)
易用性	熟练操作	A5-用户学习并熟练使用系统的时间不超过一周(M,M)
	正常操作	A6-成功操作在总操作中所占的比例超过 90%(H,M)
性能	响应时间	A7-显示一张图片的时间不超过 0.1s(H,L)
		A8-处理一张图片的时间不超过 2s(H,M)
	吞吐量	A9-数据采集的吞吐量不低于 3M/s(H,M)
	采集质量	A10-数据采集正确率不低于 80%(H,M)
可维护性		A11-维护人员遇到了响应时间和采集质量缺陷，修复该 bug，并分配 bug 修复(M,M)
可移植性		A12-移植至不同的操作系统可以稳定运行(H,M)
		A13-移植后系统性能最多降低 5%(M,M)

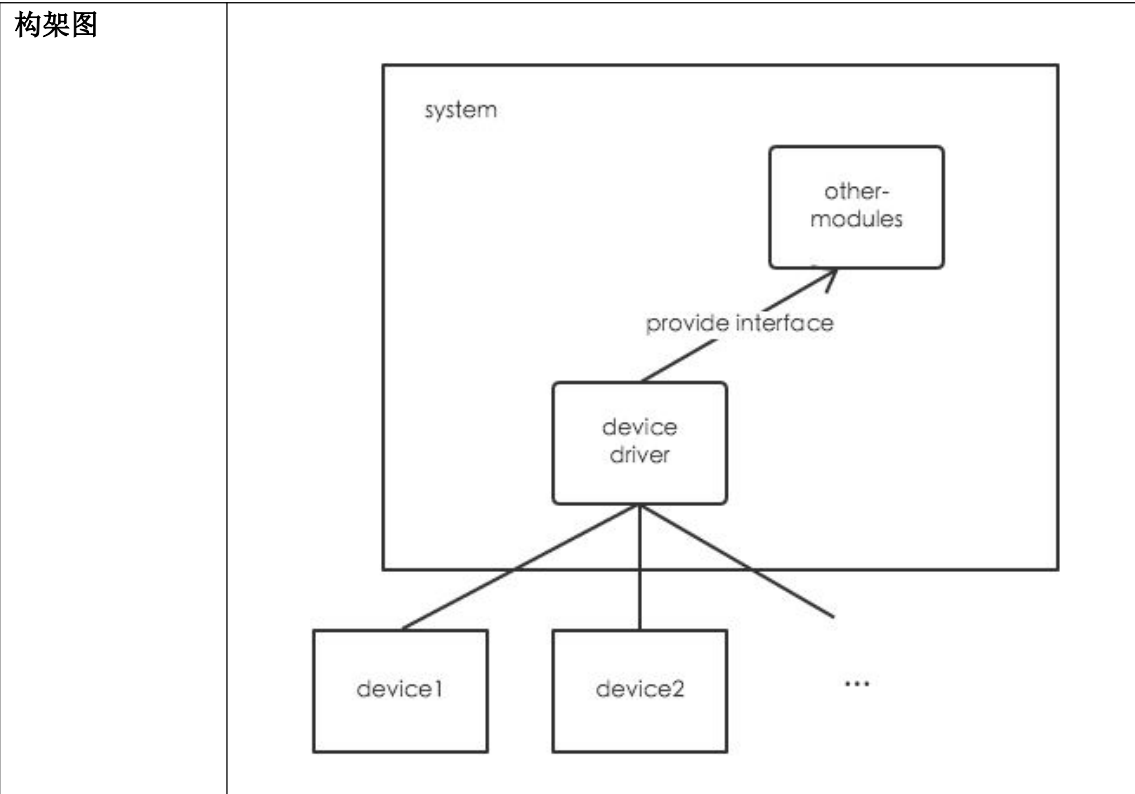
2.构架方法分析

场景号：A1	场景：添加自定义获取模式			
属性	可扩展性			
环境	正常操作			
刺激	用户请求自行定义新的图像获取模式时			
响应	添加获取模式，系统被修改或添加后，对系统其他部分影响不大。			
构架决策	敏感点	权衡点	有风险决策	无风险决策
实现相应接口	S1	T1		N1
使用外观模式	S2	T2	R1	
推理	实现相应接口并增加采集算法易于扩展，不易出错； 由于采用外观模式，将具体模式作为细粒度包装成粗粒度对象，方便新模式的添加。			

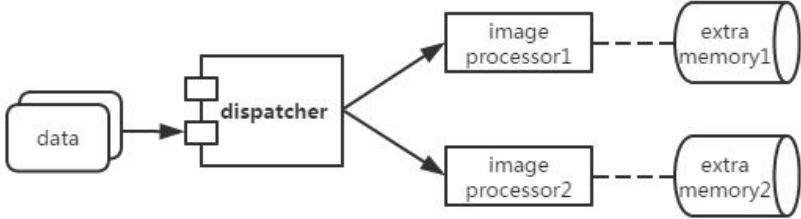
构架图



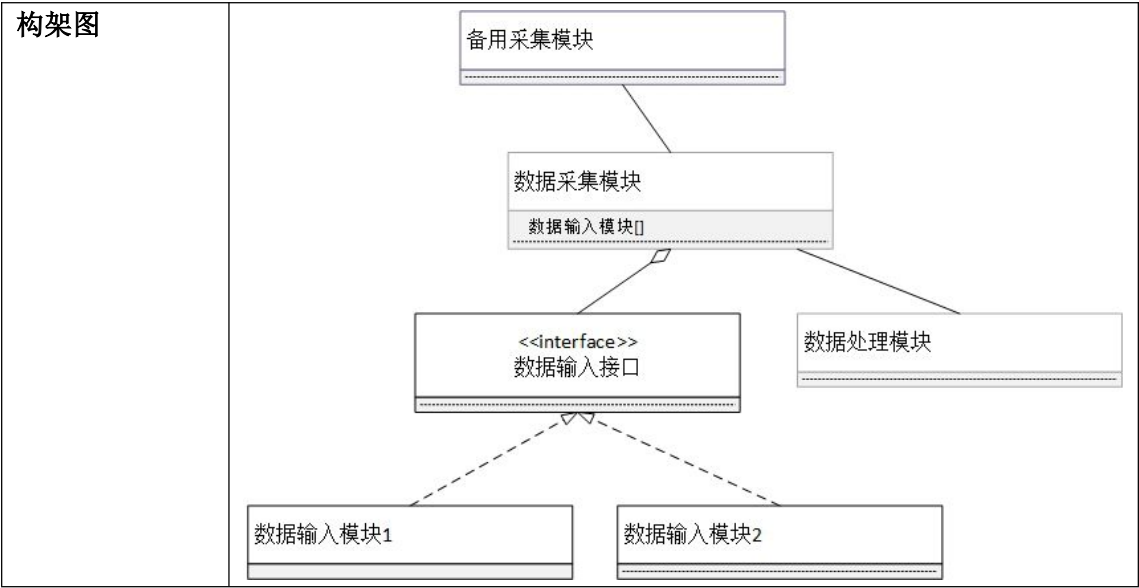
场景号：A3	场景：添加外部设备支持			
属性	可扩展性			
环境	正常操作			
刺激	用户请求使用新的外部设备时			
响应	添加对新设备的支持，添加后对系统其他部分影响不大。			
构架决策	敏感点	权衡点	有风险决策	无风险决策
封装硬件驱动代码	S3	T3		N2
预设多种外设支持	S4		R2	
推理	封装硬件驱动代码隐藏实现细节，便于新设备的添加。 提前内置对多种常见外部设备的支持。			



场景号：A4	场景：检测远程图像处理服务器故障并从中恢复过来			
属性	可用性			
环境	正常操作时需要连接到远程图像处理服务器			
刺激	远程图像处理服务器出现故障			
响应	切换至备用服务器的可用概括为 0.9			
构架决策	敏感点	权衡点	有风险决策	无风险决策
备用服务器	S5			N3
心跳	S6	T4	R3	
故障切换路由	S7			N4
推理	通过使用备用服务器来保证在服务器发生故障时，系统仍能连接到另一服务器进行数据传输。 使用心跳可以保证在 3 秒钟之内检测到服务器故障 最坏情况下，5 秒钟之内可以完成故障恢复。			
构架图	<pre>graph LR NCI((网络连接接口)) --> MS[主服务器] NCI --> BS[备用服务器] MS -.-> 心跳 (3秒) BS MS --> CS[切换服务器] BS --> CS CS --> Exit[]</pre>			

场景号：A8	场景：处理一张图片的时间不超过 2s			
属性	性能			
环境	正常操作			
刺激	用户采集到照片并传送到图像处理组件			
响应	处理图片的时间不超过 2s/张			
构架决策	敏感点	权衡点	有风险决策	无风险决策
提高计算效率	S8			N5
增加可用资源	S9		R4	
限制执行时间	S10	T5	R5	
推理	<p>提高计算速率，改进图像处理关键算法，能够有效改善系统的图像处理响应时间。</p> <p>增加可用资源虽然能够对系统性能有显著提高，但是会造成系统成本的增加。</p> <p>限制单张图片的处理时间会对系统的可用性造成一定负面影响。</p>			
构架图	 <pre> graph LR data[data] --> dispatcher[dispatcher] dispatcher --> ip1[image processor1] dispatcher --> ip2[image processor2] ip1 -.-> mem1[(extra memory1)] ip2 -.-> mem2[(extra memory2)] </pre>			

场景号：A9	场景：保证数据采集的吞吐量			
属性	性能			
环境	正常操作			
刺激	系统通过照相机或探头采集数据			
响应	数据采集的吞吐量不低于 3M/s			
构架决策	敏感点	权衡点	有风险决策	无风险决策
允许获取速度和图片质量做权衡	S11		R6	
模块化采集	S12			N6
备用采集系统	S13	T6		N7
推理	<p>在保证图片质量在一定程度之上时，可以牺牲一定图片质量来提高获取速度</p> <p>可以通过调节接入系统的输入模块的多少来调节吞吐量，当吞吐量不够时，可以向系统接入空闲的输入模块</p> <p>当当前的采集系统出现故障时，备用的采集系统保证数据采集在 2s 之内恢复且保证采集速度</p>			



场景号：A12	场景：移植至不同的操作系统			
属性	可移植性			
环境	希望将软件移植至不同的操作系统			
刺激	设计时间，部署时间，开发时间			
响应	修改与操作系统交互的代码以适应目标系统			
构架决策	敏感点	权衡点	有风险决策	无风险决策
采用分层结构	S14	T7	R7	
采用工厂模式	S15			N8
提供与操作系统进行交互的接口	S16			N9
推理	<p>由于采用分层结构，上层应用不能直接与底层的操作系统交互，耦合程度降低，同时易于修改，但是系统性能会降低。</p> <p>由于采用工厂模式，隔离了具体类的生成。</p> <p>由于提供了与操作系统进行交互的接口，具有面向接口而不是具体实现编程的优点。</p>			
构架图				

3.敏感点与权衡点列表

敏感点 ID	描述
S1	使用接口功能添加更加灵活，有利于实现可扩展性。
S2	使用外观模式将细粒度对象包装成粗粒度对象，有利于新模式的增加，提高了可扩展性。
S3	封装硬件驱动代码消除了数据与操作分离带来的问题，有利于新设备添加，实现可扩展性。
S4	预先提供对 4 种外部设备的支持。
S5	备用服务器的使用可以在系统发现服务器故障后，作为主服务器的代理，在服务器恢复之前进行工作，有利于系统的可用性。
S6	每隔 3 秒向服务器发送一次心跳，可能会占用主服务器的 CPU 资源，对性能造成负面影响。
S7	使用故障切换路由来保证在检测到服务器故障时能够迅速进行备用服务器的路由切换，有利于可用性的增加。
S8	改进图像处理的算法，进而提高计算效率，有利于提高系统的响应时间。
S9	增加可用资源，采用速度更快的处理器，并在成本允许的情况下增加额外的处理器和内存，从而进行图形的并发处理来减少图像处理的等待时间，提高系统性能。
S10	限制单张图片的处理时间，当超过 2s 还未完成处理，则停止对该图片的处理，并记录日志。
S11	允许获取速度和图片质量做权衡，将数据采集与数据处理分成两个模块，实现权衡
S12	通过提供数据输入的接口，一个数据采集模块可以拥有多个数据输入通道，不同的通道可以有不同实现方式和采集速度，利于调节吞吐量
S13	使用备用采集系统可以在原采集系统出现问题时替代
S14	采用分层结构可避免与底层操作系统的直接交互，有利于实现可移植性。
S15	采用工厂模式可隔离具体类的生成，与直接与操作系统交互的代码相隔离，提高了可移植性。
S15	通过提供接口的方式，避免了与具体实现交互，降低耦合度，提高可移植性。

权衡点 ID	描述
T1	实现相应接口在提高可扩展性的同时，也实现了可维护性。
T2	外观模式在提高可扩展性的同时，也可能会影响性能。
T3	封装硬件驱动代码在实现可扩展性的前提下，也提高了可维护性。
T4	使用心跳策略有利于系统及时发现服务器的故障，对系统的可用性有正面作用；但是会占用服务器的 CPU 资源，性能产生负面影响。
T5	限制单张图片的处理执行时间有利于保证其他图片处理的性能，但是会对系统的可用性造成一定的负面影响。
T6	使用备用采集系统需要大量数据备份，可能影响性能

T7	采用分层结构除了将对可移植性造成影响之外，还可能会影响性能，可扩展性，可修改性，可维护性等等。
----	---

4.有风险决策与无风险决策列表

有风险决策 ID	描述
R1	使用外观模式添加新的子系统时可能需要修改外观类或客户端的源码，违背了开闭原则。
R2	预先设置对多种设备的支持可能造成冗余。
R3	不采用每隔 3 秒向服务器发送一次心跳的策略，因为这样会占用服务器 CPU 资源，继而影响服务器的性能，可能会使服务器的响应时间和吞吐量收到影响。
R4	不采用增加可用资源。虽然提高了系统的性能，但是会增加系统的成本。
R5	不采用限制执行时间，因为虽然这样有利于系统的性能提高，但是会对系统的可用性造成负面影响。
R6	当吞吐量过量时，图片质量会受到极大影响
R7	分层结构可降低软件的耦合度，提高可移植性，但是可能会造成系统整体性能的降低。

无风险决策 ID	描述
N1	使用接口有利于实现可扩展性。
N2	封装硬件驱动代码降低了复杂度，易于扩展。
N3	使用备用服务器可以在系统发现服务器故障后，作为主服务器的代理，在服务器恢复之前进行工作，有利于系统的可用性。此决策主要针对可用性的恢复方面进行对可用性的提升。
N4	使用故障路由切换的策略可以在系统发现服务器故障时快速进行备用服务器的切换，有利于系统的可用性。
N5	改进处理图像的关键算法，提高计算效率，从而能够减少系统的响应时间，提高系统的性能。
N6	模块化采集隔离了数据输入的实现和使用，有利于使用不同方式增加数据吞吐量
N7	备用采集系统与原采集系统拥有相同的数据备份，当现有采集系统不工作时，备用系统可以在不丢失数据的情况下继续接收并传输数据
N8	工厂模式隔离了具体类的生成，提高了可移植性。
N9	提供与操作系统进行交互的接口降低了耦合度。

5.个人总结

学号：131250185 姓名：倪小凡

本次作业我们小组一起学习 ATAM，首先画出了 utility tree，决定了需要讨论的 6 个场景。其中我负责添加自定义获取模式和添加外部设备的讨论，列出了相关的敏感点，权衡点，有风险决策和无风险决策。

学号：131250168 姓名：吴超月

用 ATAM 的体验：ATAM 是评估软件构架的一种综合全面的方法，这种方法不仅可以揭示出构架满足特定质量目标的情况，而且可以使我们更清楚地认识到质量目标之间的联系，即如何权衡诸多质量目标。在使用 ATAM 方法进行架构评估的过程中，我体会到使用 ATAM 的主要目的在于在分析属性的架构决策时找到他们的敏感点和权衡点，这样才能在各种质量属性和架构决策的关系中作出最正确、全面的决定。

个人贡献：我们小组四人首先一起进行了质量属性效用树的构建，然后分配场景到个人进行构架方法分析和敏感点、权衡点、有/无风险决策的编写。我所负责的是可用性和性能中两个场景的构架方法分析和相关敏感点、权衡点、有/无风险决策的编写，并负责最终的文档整合。

学号：131250177 姓名：罗瑶

在这次作业中，我们一起查阅资料，共同完成了 utility tree 的定义，并且讨论出需要进一步讨论的 6 个场景。之后我负责场景保证数据采集的吞吐量的讨论。列出了相关的敏感点，权衡点，有风险决策和无风险决策，完成了此次作业。

学号：131250158 姓名：邹卓晋

在此次作业中，我们小组首先一起学习了 ATAM，然后在上次 ADD 作业的基础上画出了 utility tree，并从中找出了相对而言比较重要的 6 个场景。之后我负责移植至新系统场景的分析与文档编写，并列出了其中的敏感点，权衡点，有风险决策和无风险决策。通过对 ATAM 的学习，我对于怎样在各个架构决策之间进行权衡分析有了更为深刻的认识。